# Architectural Overview Of The Common Language Runtime

Jim Miller
Lead Program Manager
Runtime Kernel
Microsoft Corporation

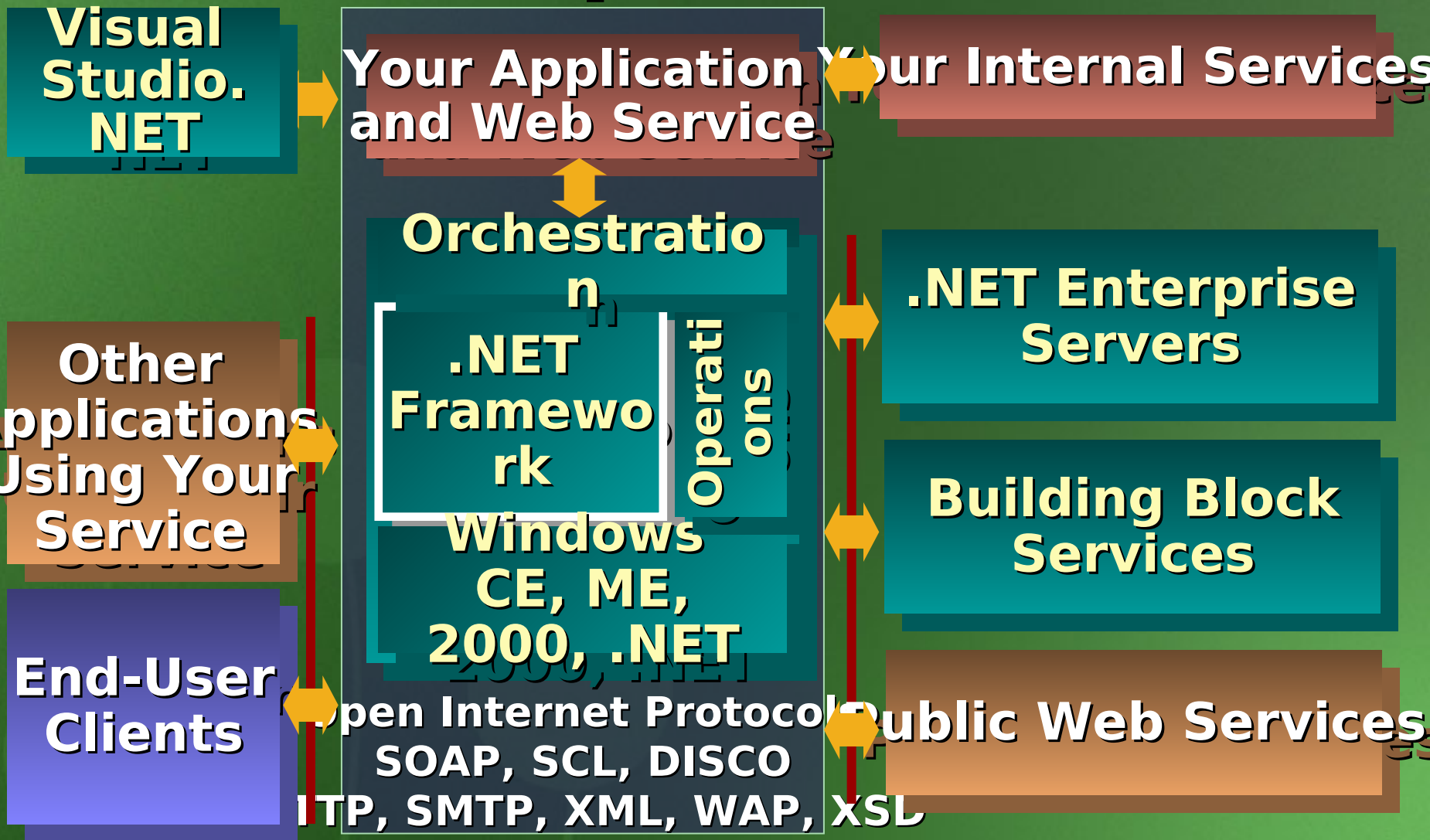
3-313


Microsoft®
PDC 2000
Professional Developers Conference
Microsoft®
.net™
the defining point

# Outline

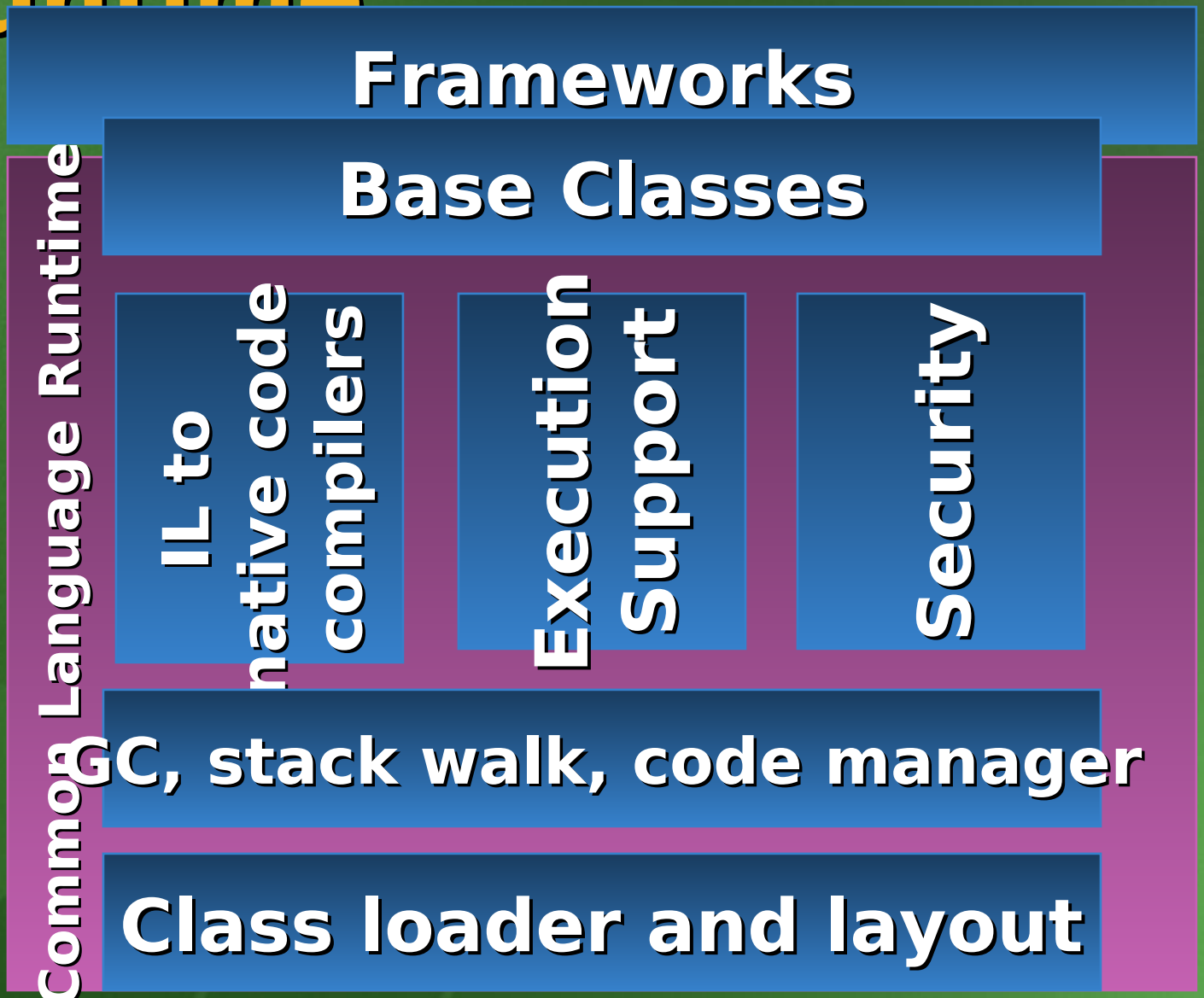
- .NET Framework
- Common Language Runtime Goals
- Design and Compile Time
- Deployment Time
- Execution Time
- Summary

# .NET Blueprint



# .NET Framework

# Common Language Runtime



# Outline

- .NET Framework
- Common Language Runtime Goals
- Design and Compile Time
- Deployment Time
- Execution Time
- Summary

# Simpler Development

- Write less, reuse more
  - Broad, consistent framework
  - Classes as well as interfaces
- Plumbing disappears
  - Metadata
  - Transparent proxies
  - Memory management
- Great WYSIWYG tool support
  - Designers and wizards
  - Debuggers
  - Profilers

**You get increased productivity!**

# Simpler, Safer Deployment

- No registration, zero impact install
  - XCOPY deployment, incremental download
- Side-by-side versions of shared components
  - Capture version at compile time
  - Administrative policy at run time
- Evidence-based security policy
  - Based on code as well as user
  - Code origin (location)
  - Publisher (public key)

You get end of DLL hell!

# Scalability

- Smart device to Web farm
- Automatic memory management
  - Self-configuring
  - Dynamically tuning
- Thread pool
- Asynchronous messaging
  - Object remoting
  - Events
- Smart device version
  - Multiple RTOSes
  - Same tools used for desktop

**You get better performance for less cost!**

# Rich Web Clients, Safe Web Hosting

- Win Forms on the client
- ASP+ Web Forms on the server
- Code is granted permissions
  - Evidence is used by policy to grant permissions
- Application that starts runtime
  - Like Internet Explorer, IIS, SQL Server™, Shell
  - Provides some evidence
  - Controls code loading
  - Maps applications to processes

You get robust applications!

# Convergence Of Programming Models

- COM, ASP, Data
  - All services available
  - Many services redesigned
    - Ease of use
    - Scalability
    - Consistent API
- Consistent framework raises the abstraction layer
- Gradual transition from simplicity to full power

You get less training, higher productivity!

# Multiple Languages

- Common type system
  - Object-oriented in flavor
  - Procedural languages well-supported
  - Functional languages possible
- CLS guides frameworks design
  - Rules for wide reach
  - All .NET Framework functionality available
- Over 15 languages investigated
  - Most are CLS consumers
  - Many are CLS extenders

**You get choice of tool for the job!**

# Review

- Multi-language, secure, mobile code
- Scalable from server to small device
- Side-by-side versioning
- Self-description throughout
- Rich frameworks
- Simplified programming model
- Strong, integrated tools support

You get productivity, simplicity, reliability!

# Outline

- .NET Framework
- Common Language Runtime Goals
- Design and Compile Time
- Deployment Time
- Execution Time
- Summary

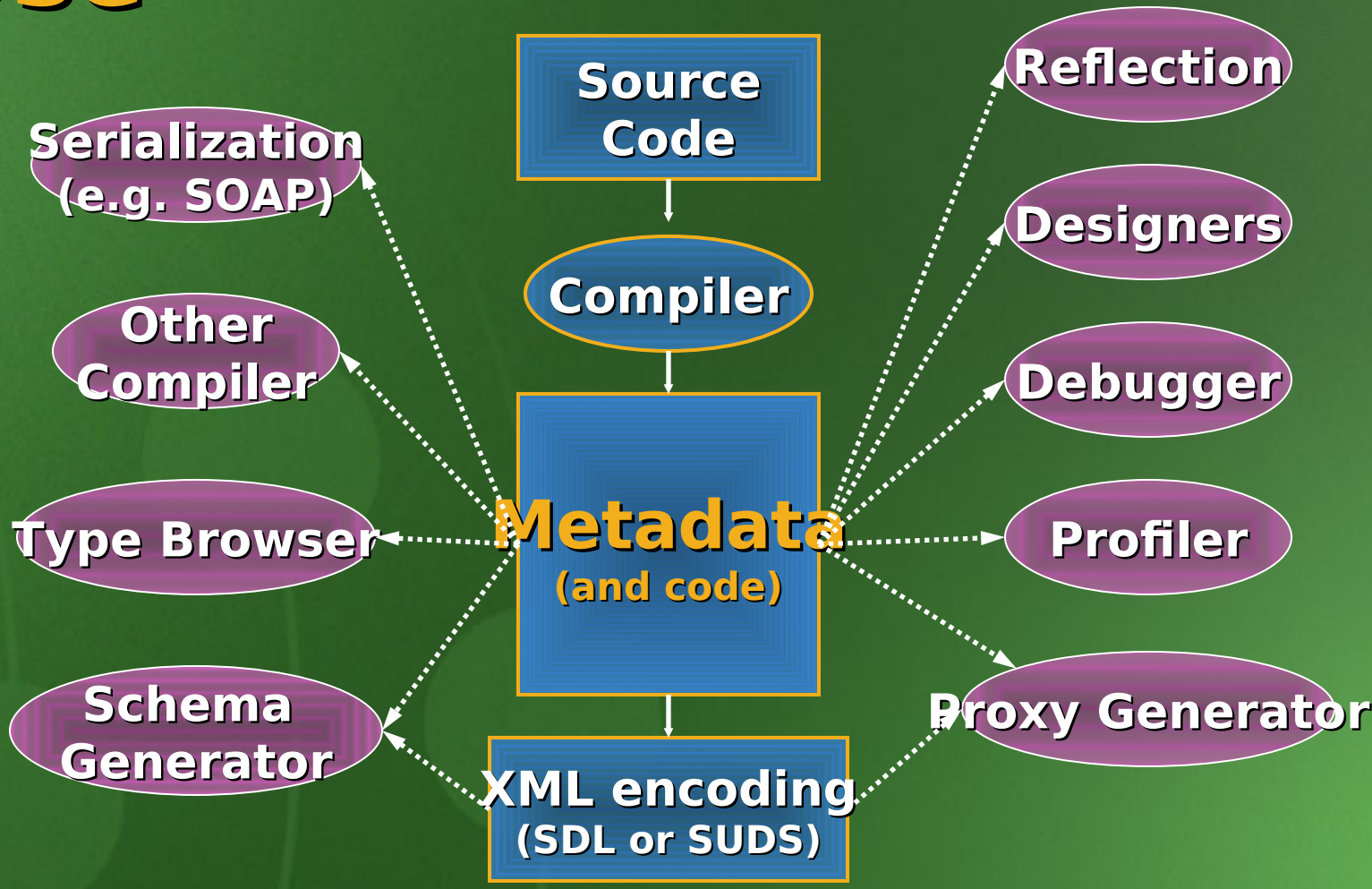
# Metadata

- Key to simpler programming model
- Generated automatically
  - Stored with code in executable file (.dll or .exe)
  - Uses existing COFF format
    - Via existing extension mechanism
  - Stored in binary format
- Convertible to/from XML Schema
- Convertible to/from COM type

# What's In The Metadata

- Description of deployment unit (assembly)
  - Identity:  name, version, culture[, public key]
  - What types are exported
  - What other assemblies it depends on
  - Security permissions needed to run
- Description of types
  - Name, visibility, base class, interfaces implemented
  - Members (methods, fields, properties, events, nested types)
- Custom attributes
  - User-defined
  - Compiler-defined
  - Framework-defined

# Metadata: Creation And Use



# Compilers Use Metadata

- For cross-language data type import
- Emit metadata with output code
  - Describe types defined and used
  - Record external assemblies referenced
  - Record version information
- Custom attributes can be used
  - Obsolete
  - CLS compliance
  - Compiled for debugging

# Other Tools Use Metadata

- Designer behavior
  - Controlled by user-supplied attributes
    - Category
    - Description
- Designer extensibility
  - User-supplied attributes specify code to use
    - Type converters
    - Editors
- Web methods marked by custom attribute
- Type viewer

# Outline

- .NET Framework
- Common Language Runtime Goals
- Design and Compile Time
- Deployment Time
- Execution Time
- Summary

# Assemblies

- Unit of deployment
  - One or more files, independent of packaging
  - Self-describing via metadata (“manifest”)
- Versioning
  - Captured by compiler
  - Policy per-application as well as per-machine
- Security boundary
  - Assemblies are granted permissions
  - Methods can demand proof that a permission has been granted to entire call chain

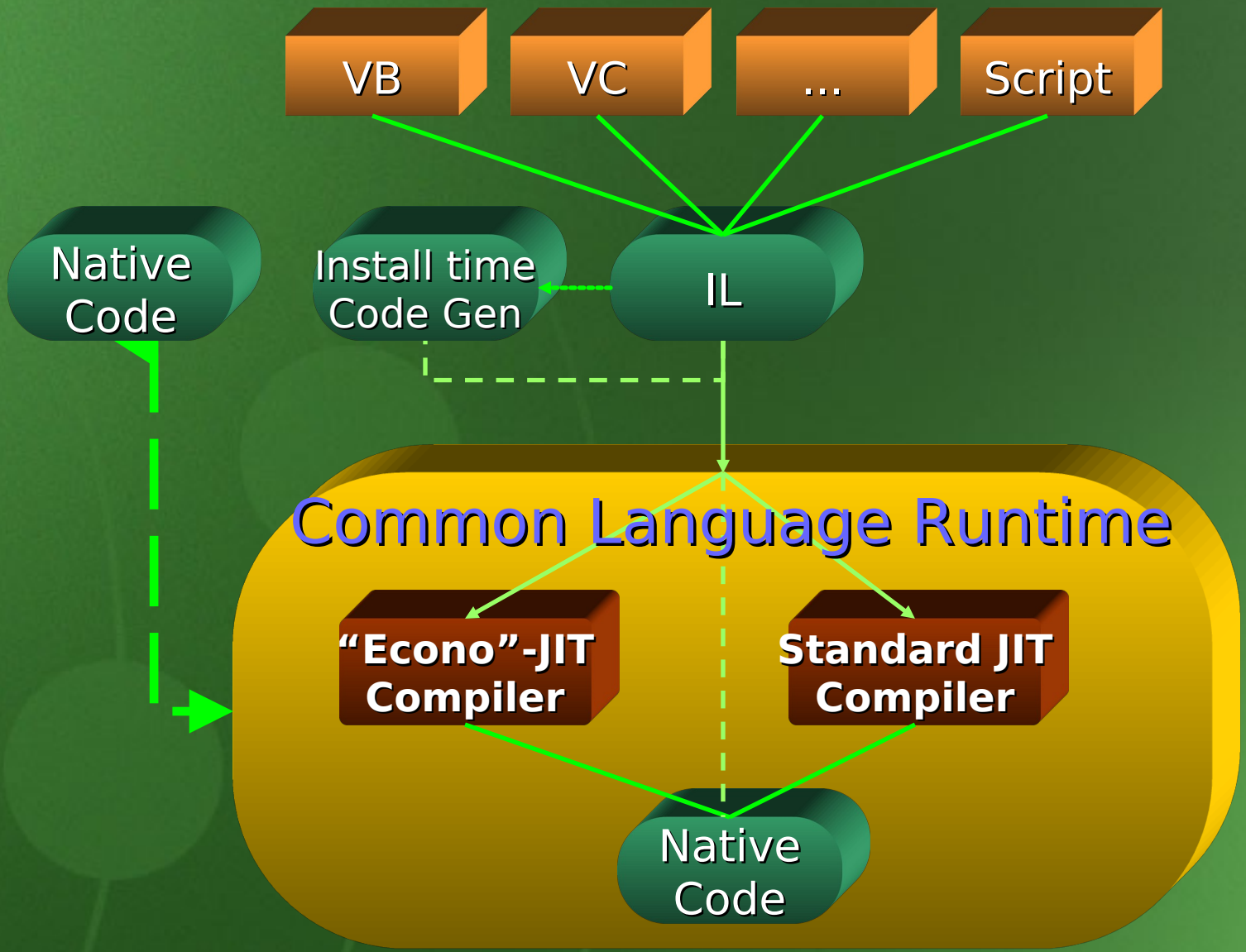
# Applications

- Applications are configurable units
  - One or more assemblies
  - Application-specific files or data
- Assemblies are located based on…
  - Their logical name and
  - The application that loads them
- Applications can have private versions of assemblies
  - Private version preferred over

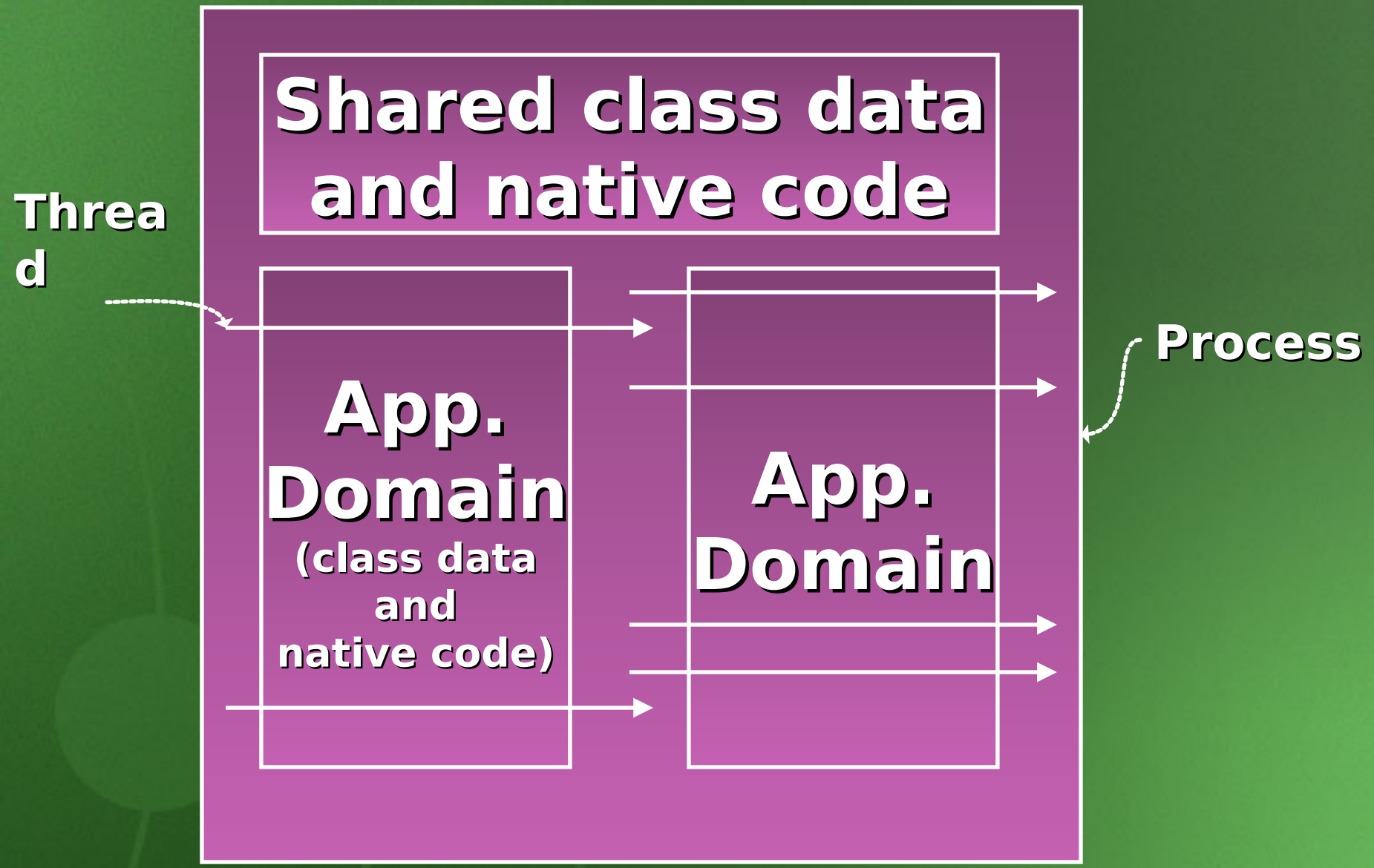
# Outline

- .NET Framework
- Common Language Runtime Goals
- Design and Compile Time
- Deployment Time
- Execution Time
- Summary

# Execution Model



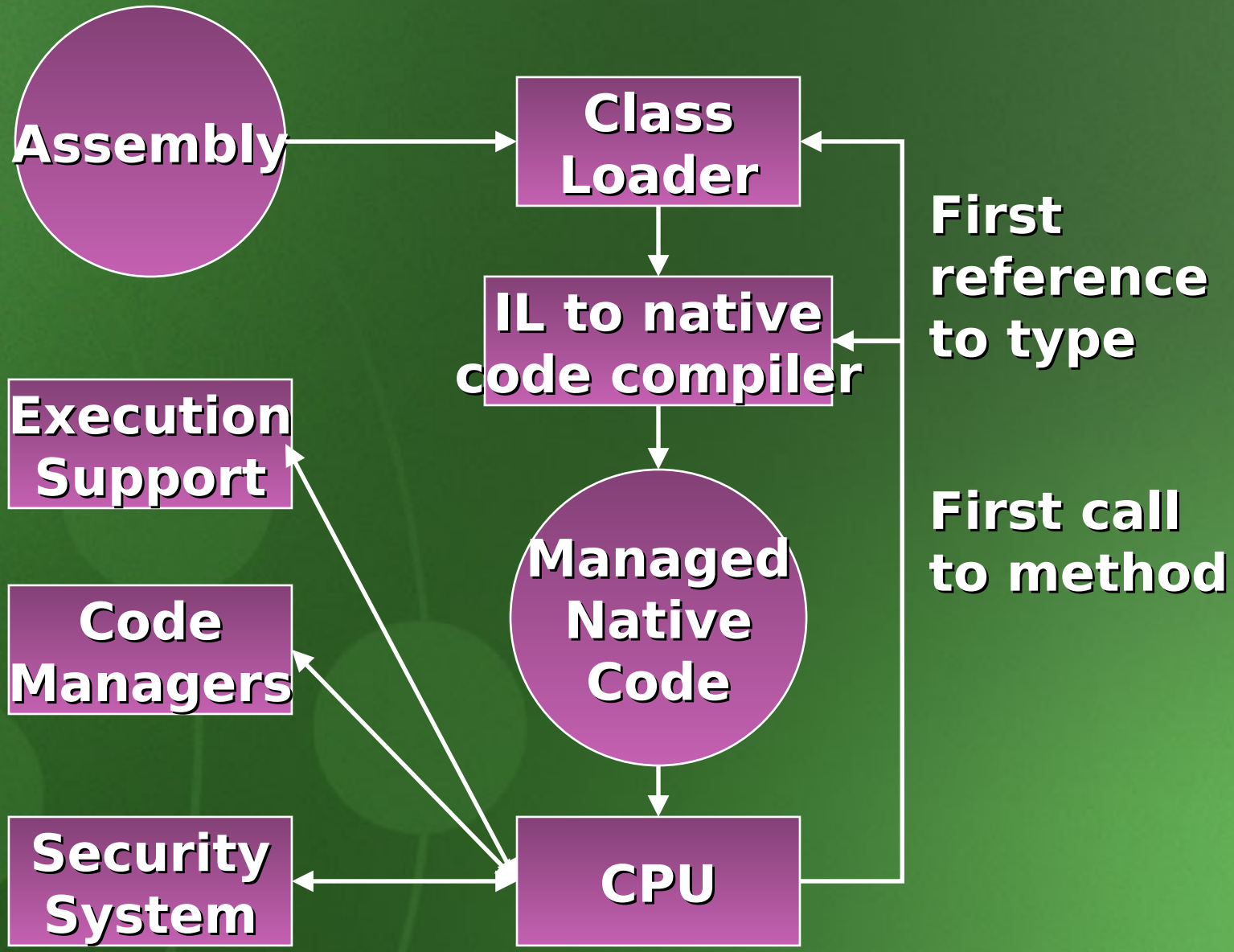
# The Process Model



# Managed Code

- Managed code provides…
  - Metadata describing data
  - Location of references to objects
  - Exception handling tables
- So runtime can provide…
  - Exception handling
  - Security
  - Automatic lifetime management
  - Debugging and profiling

# Runtime Control Flow



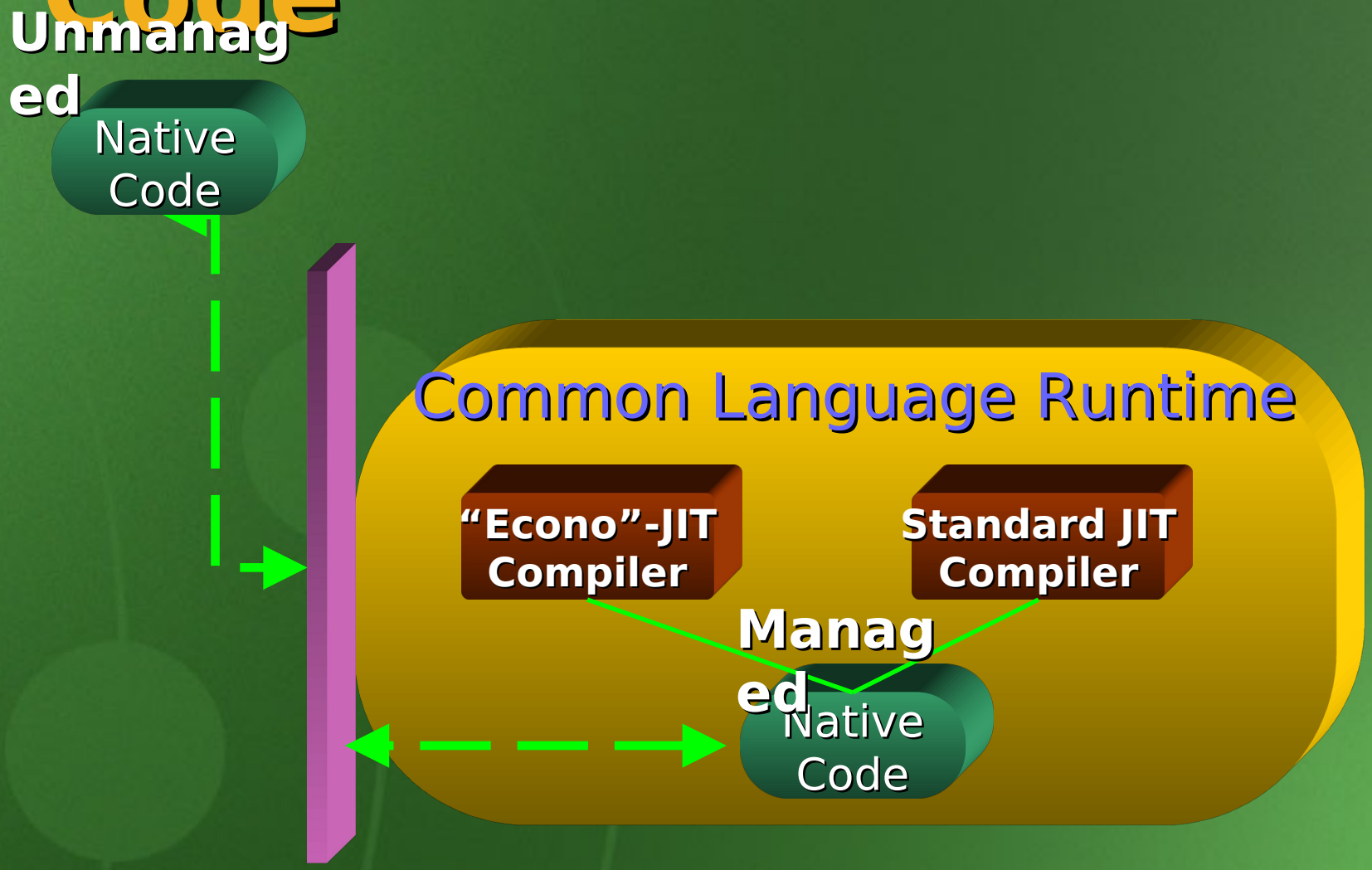
# Compiling IL To Native

- “Econo” JIT
  - Generates unoptimized native code
  - Code can be discarded and regenerated
- “Standard” JIT
  - Generates optimized native code
  - Includes verification of IL code
- Install time code generation
  - Done at install time
  - Reduces start-up time
  - Native code has version checks and reverts to runtime JIT if they fail

# Managed Data

- Layout Provided by Runtime
  - Usually automatic
  - Metadata can specify
    - Order
    - Packing
    - Explicit layout
- Lifetime Managed by Runtime (GC)
  - Working set is compacted
  - Data is moved
  - Object references are updated
  - No more intrusive than a page fault

# Calling Unmanaged Code



# Crossing The Boundary

- Mode transition for code manager
  - Calling conventions differ on x86
  - Fast, rarely more than register shuffle
- Data type marshalling
  - Representations may not be the same
  - Pinning, copying, and/or reformatting needed
  - Custom marshalling supported
- The IL to native compilers help
  - In-line code transition and simple marshalling
  - Per call site cost is very low
    - Plus a small cost on entry to a procedure that can make calls across

# Three Mechanisms

- Via function pointer
  - No mode transition, no marshalling
  - Stringent restrictions on called code
- “It just works”
  - Mode transition, no marshalling
  - All work done by tools (VC)
    - Linker resolves references
    - Class loader creates transition thunks
- Platform invoke (P/Invoke)

# COM Interop

- Uses P/Invoke mechanism for calls
- Object identity is maintained (IUnknown)
- Selected COM interfaces automatically created
- Types are automatically exported to COM and registered
- COM type libraries can be converted to metadata and imported
- Many data types are automatically marshaled

# Outline

- .NET Framework
- Common Language Runtime Goals
- Design and Compile Time
- Deployment Time
- Execution Time
- Summary

# Summary

- Simpler…
  - Development, deployment, administration
- Multi-language, secure, mobile code
- All code compiled before execution
  - Not your traditional virtual machine!
- Full interoperation with unmanaged code
  - COM, COM+, Web services, Win32, your DLLs

You get a strong foundation for the future!

# Related Sessions And References

- For next level of detail
  - 3-314 (type system)
  - 3-215 (deployment)
  - 3-211 (COM perspective)
  - 3-222 (frameworks)
  - 3-344 (security)
- For all the details, see the SDK
  - Technical Overview, under Developers Specifications in the Documentation
  - Tool Developers' Guide

# Questions?

Where do you want to go today?

Microsoft